Data Mining

데이터 시각화 (Visualization)

학습 목표

- 데이터를 시각화 하는 목적과 주요 시각화 라이브러리인 Maplotlib의 사용법을 알아보고, 몇 가지 시각화 예제를 살펴본다.

주요 내용

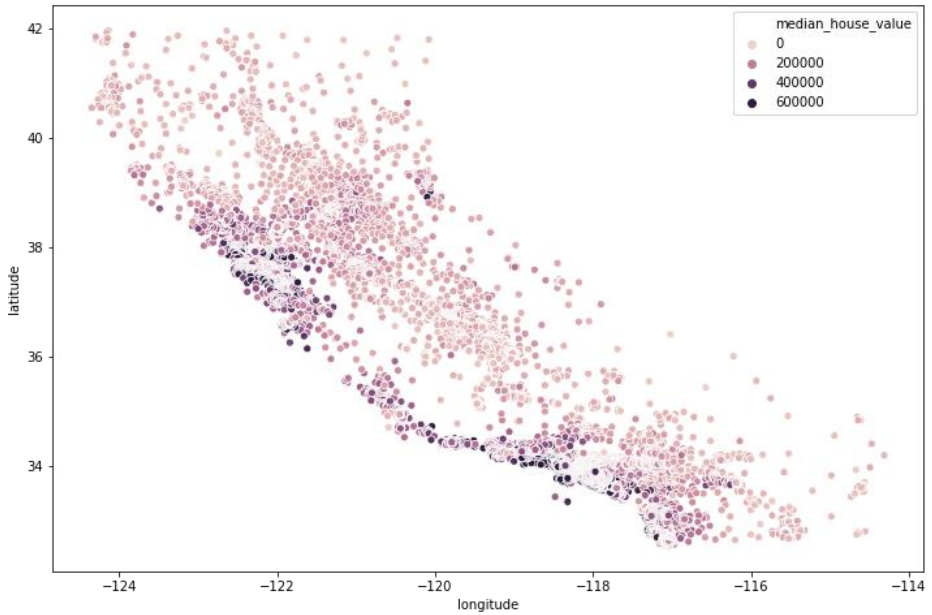
1. 데이터 시각화
2. Matplotlib
3. 예시

1. 데이터 시각화

시각화 (Visualization)

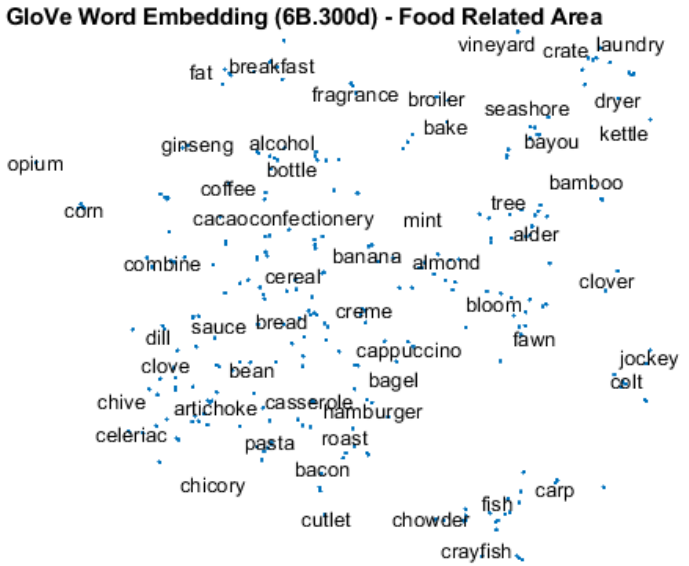
시각화의 목적은 데이터를 탐색하거나 마이닝 결과를 전달하는데 있다.

데이터 탐색 (exploration)



- 캘리포니아 주택 가격 데이터 시각화

마이닝 결과 전달 (communication)



- 단어 임베딩 (word embedding) 시각화

주요 시각화 방식

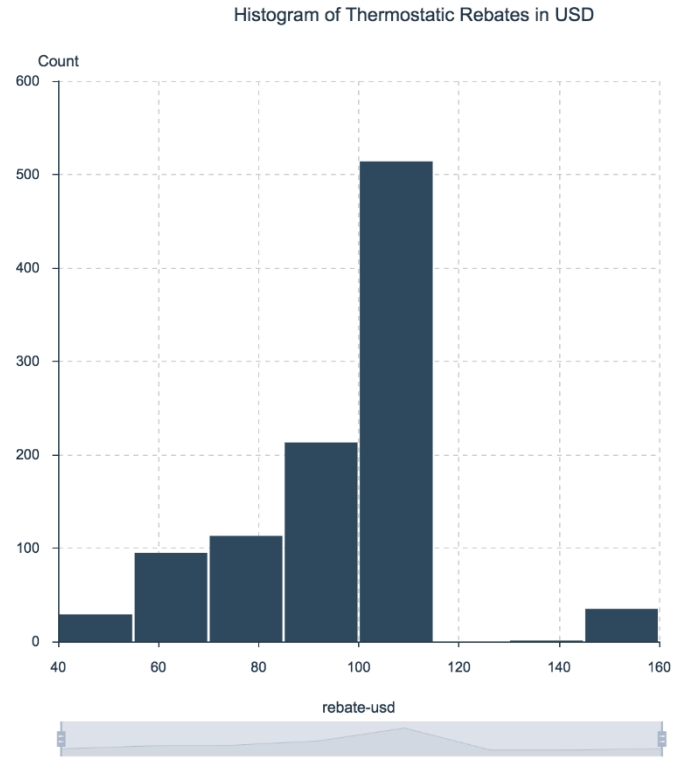
10 Visualizations Every Data Scientist Should Know

1. **히스토그램** (Histograms)
2. **막대/파이 차트** (Bar/Pie charts)
3. **산점도/직선 그래프** (Scatter/Line plots)
4. **시계열 그래프** (Time series plots)
5. **관계 맵** (Relationship maps)
6. **히트 맵** (Heat maps)
7. **지도** (Geo Maps)
8. **3D 그래프** (3-D Plots)
9. **고차원 그래프** (Higher-Dimensional Plots)
10. **단어 클라우드** (Word clouds)

https://www.datasciencecentral.com/profiles/blogs/10-visualizations-every-data-scientist-should-know

히스토그램 (Histograms)

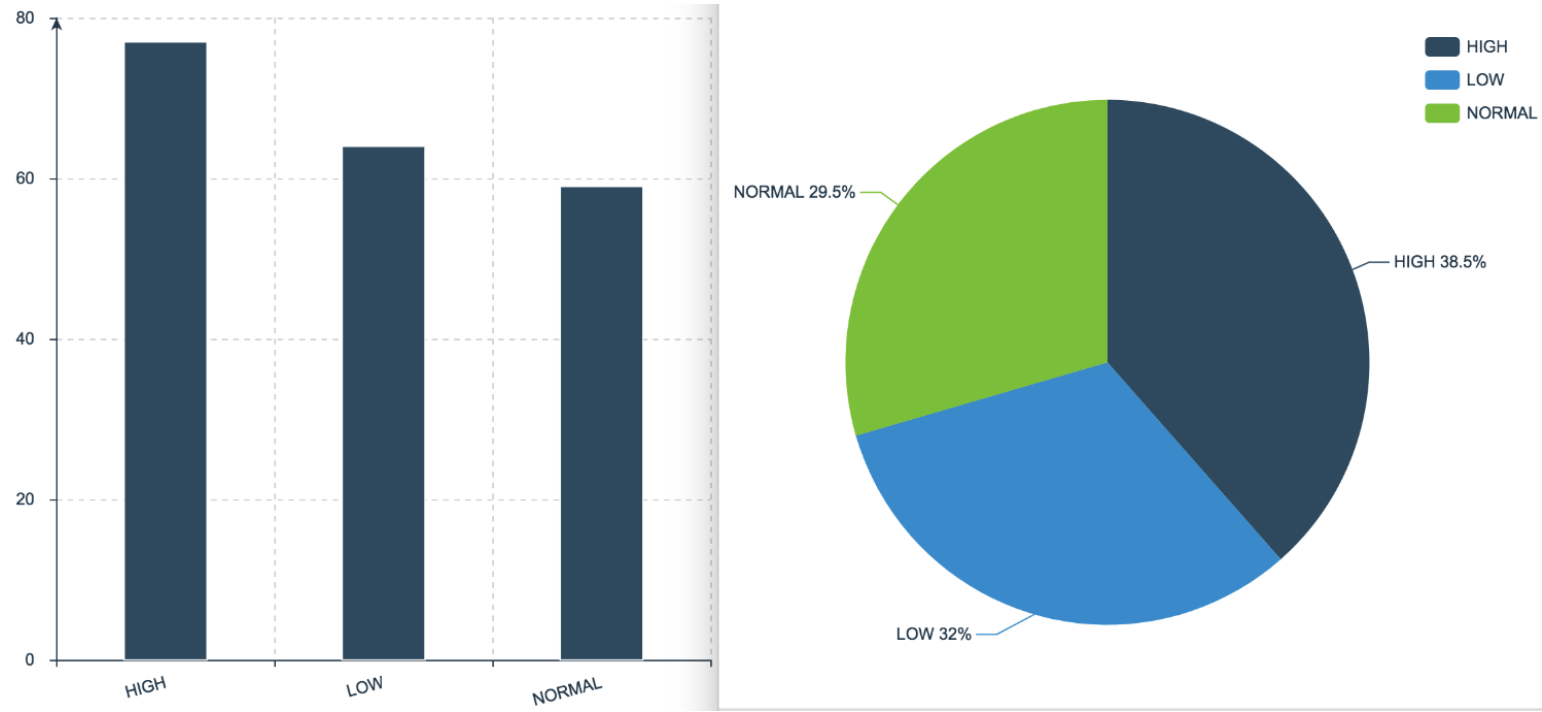
지능형 온도 조절 장치의 할인액



- 수치형 데이터의 분포를 확인하기에 유용

막대/파이 차트 (Bar/Pie charts)

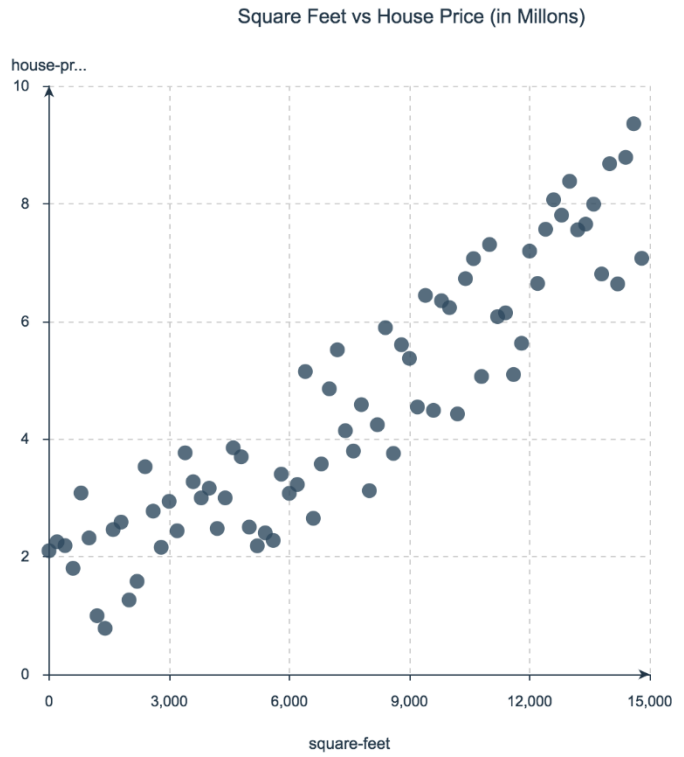
환자의 혈압을 HIGH, NORMAL, LOW로 구분한 챠트



- 명목형 데이터를 확인할 때 유용
- 단, 카테고리가 많으면 시각화에 방해가 되므로 Top N을 뽑아서 시각화 한다.
- 파이 차트는 파이의 크기가 잘 구분이 되지 않으므로 주의할 것

산점도/직선 그래프 (Scatter/Line plots)

집 값과 평방 피트 간의 관계



- 두 변수의 관계를 확인할 때 유용

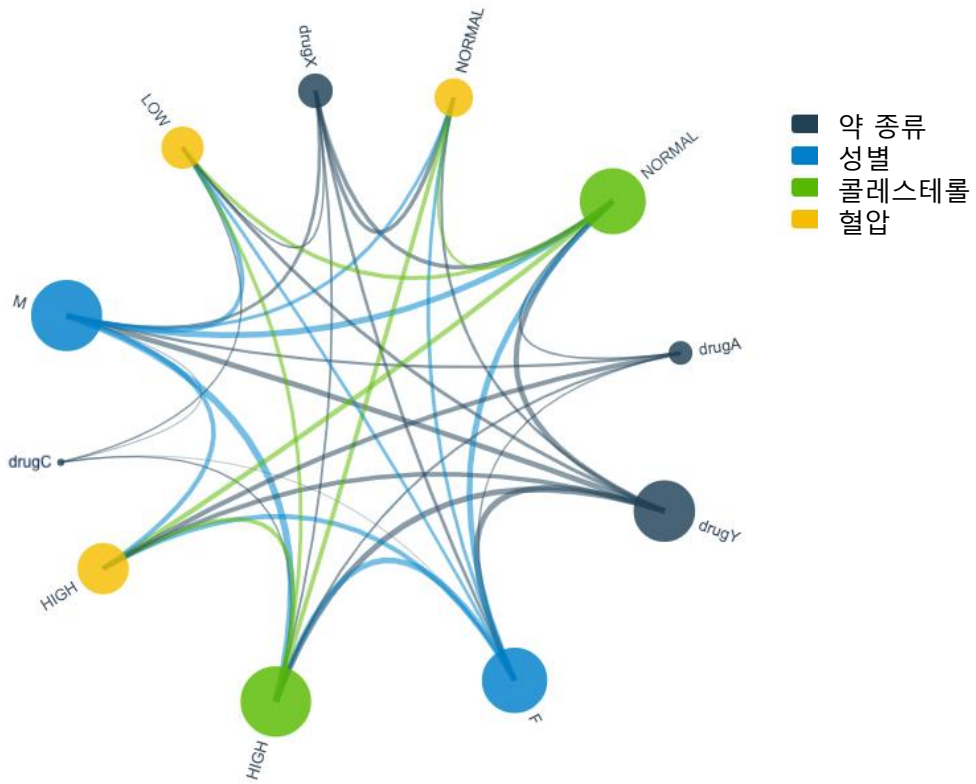
시계열 그래프 (Time series plot)

2015년에서 2017년 사이에 테슬라 주식 일일 마감 가

- 시간에 따라 변수가 변화하는 트렌드를 분석하기 위한 용도

관계 맵 (Relationship maps)

약 처방과 환자의 진단 관계



- 복잡한 가정을 수립할 때 데이터의 관계를 시각화 하는 방법
- 컬럼 별로 다른 색으로 표시 됨
- 선의 두께는 두 컬럼 사이에 관계의 중요도

의사에게 약 처방 가이드를 제시하고자 함

약 처방과 환자의 진단 상태 데이터셋의 관계

- 약 A, B, X, Y에 대해 환자 별로 하나의 약을 처방
- 각 환자의 진단 기록을 데이터 셋으로 관리

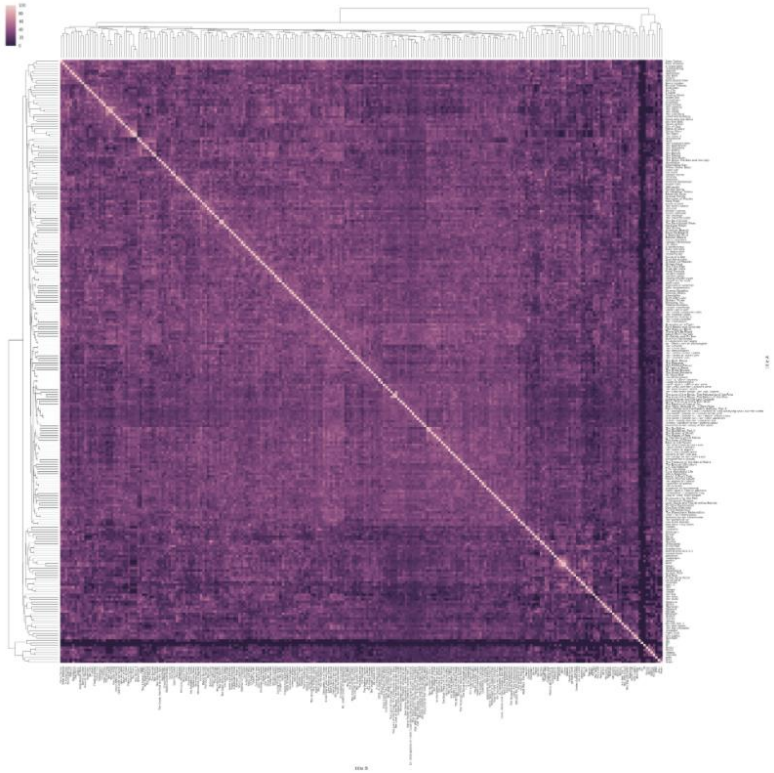
다음과 같은 사실을 알 수 있다.

- 고혈압 환자는 A 약을 처방 받음
- 저혈압이면서 콜레스테롤이 높은 환자는 C 약을 처방 받음
- 약 X를 처방 받은 환자는 고혈압 증상을 보이지 않음

연구 분야를 선정하거나, 약의 용도를 예측할 때 사용

히트 맵 (Heat maps)

IMDB 영화 제목 간의 레벤슈타인(Levenshtein) 거리

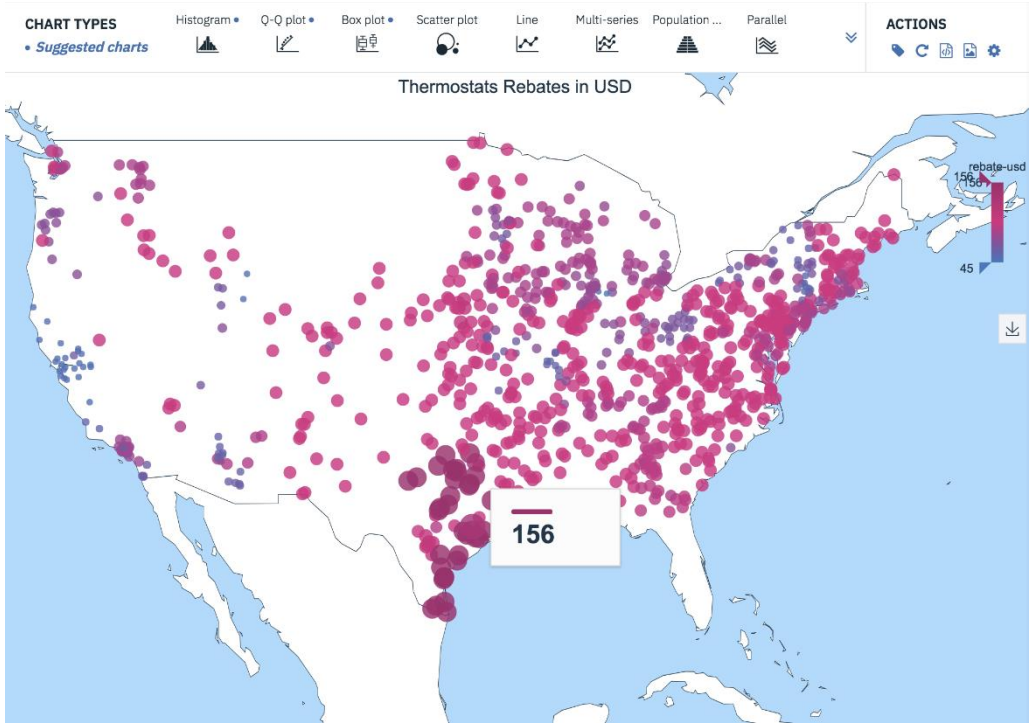


- 색으로 데이터의 빈도나 트렌드를 파악

- 제목 간에 거리가 멀수록 어두워 짐
- *Superman*은 *Batman Forever*와 멀지만 *Superman 2*와는 가까움
- **레벤슈타인**(Levenshtein) 거리 : 단어를 다른 단어를 바꿀 때 최소 글자 편집 수

지도 (Geo Maps)

지역 별 지능형 온도 조절 장치의 할인액



- 지역에 따른 변수의 변화를 파악할 수 있음

- 할인액이 낮으면 파란색, 높으면 빨간색으로 표기
- 위도 경도 외에 우편번호, 지역 코드, 카운티 데이터, 공항 데이터 등에 따라 다양한 문맥을 제공할 수 있음

단어 클라우드 (Word clouds)

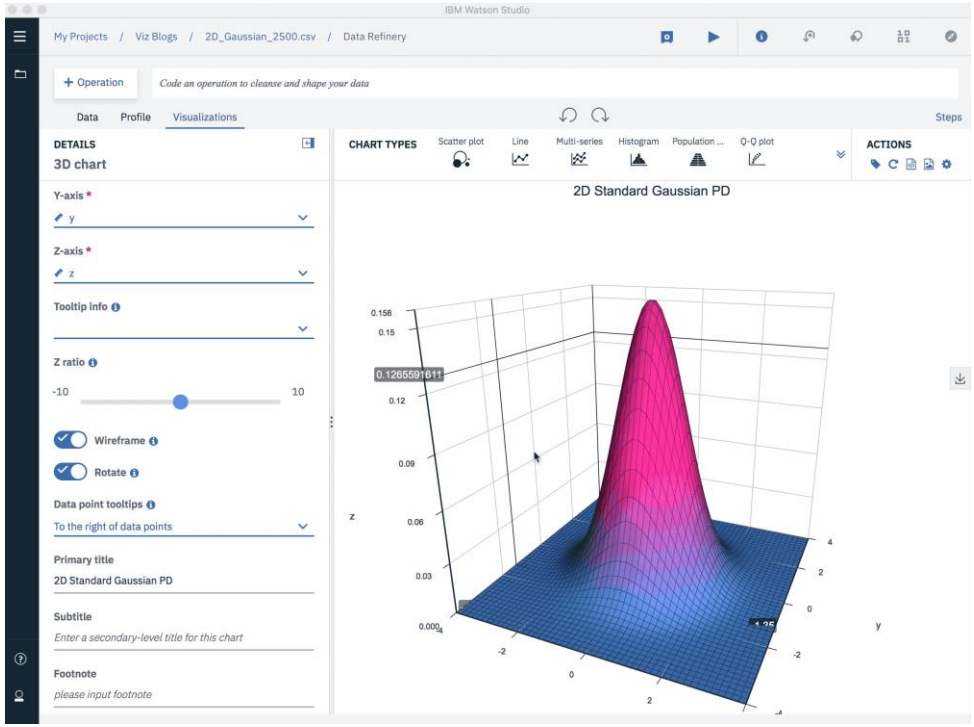
영화에 대한 긍정 리뷰 단어

영화에 대한 부정 리뷰 단어

- 단어의 빈도가 클수록 글자의 크기가 커짐

3D 그래프 (3D Plots), 고차원 그래프 (Higher-Dimensional Plots)

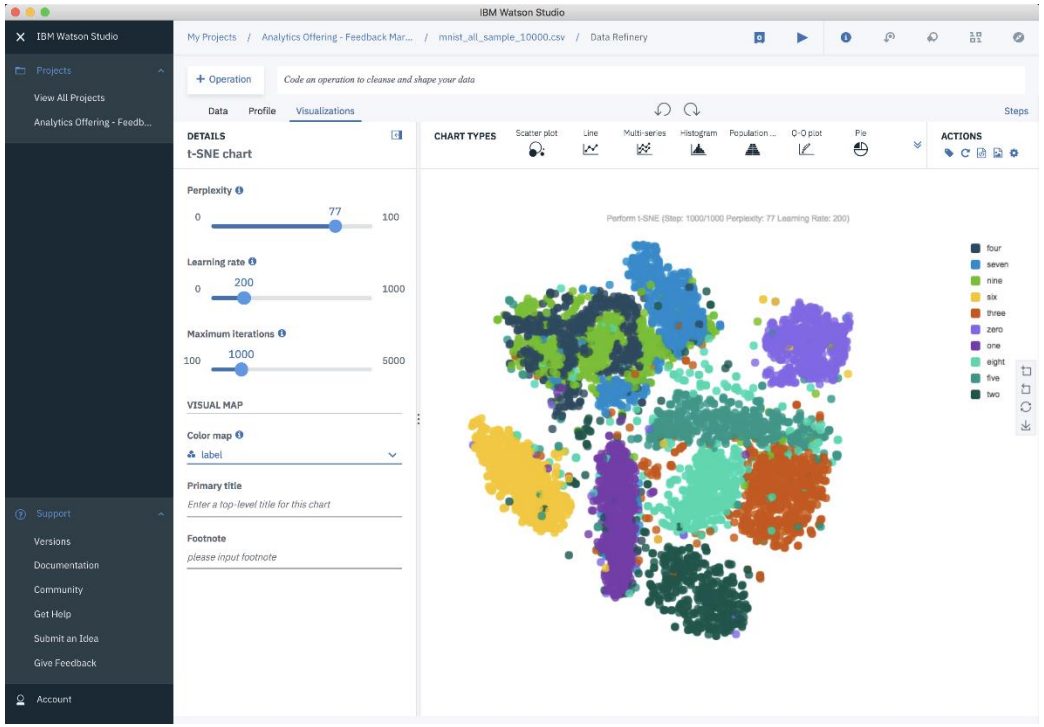
2-차원 가우시안 확률 분포



- 3D로 표현했을 때 interactive하게 관찰

MNIST 필기체 숫자 2차원 시각화 (t-SNE)

t-Stochastic Neighbor Embedding (t-SNE)



- 고차원 데이터의 경우 차원 축소를 한 후에 시각화

2. Matplotlib

Matplotlib

Matplotlib는 가장 많이 사용되고 있는 시각화 라이브러리이다.

Matplotlib는 MATLAB과 비슷하게 데이터를 그리기 위한 파이썬 그래프 함수 라이브러리

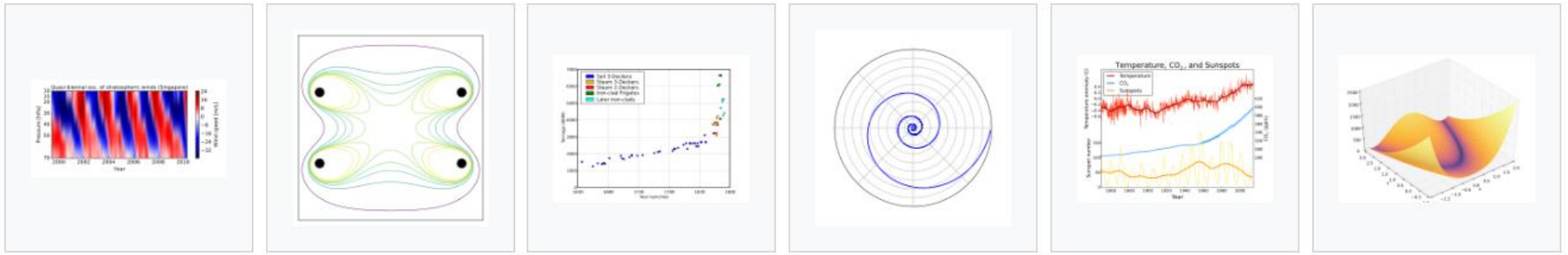


Image plot | Contour plot | Scatter plot | Polar plot | Line plot | 3-D plot

https://matplotlib.org/2.0.2/gallery.html

matplotlib

일반적으로 별명은 plt로 사용

```
from matplotlib import pyplot as plt
```

시각화를 할 때는 matplotlib의 pyplot을 사용

X좌표 y좌표

```
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')
```

선 그래프 그리기

```
plt.show()
```

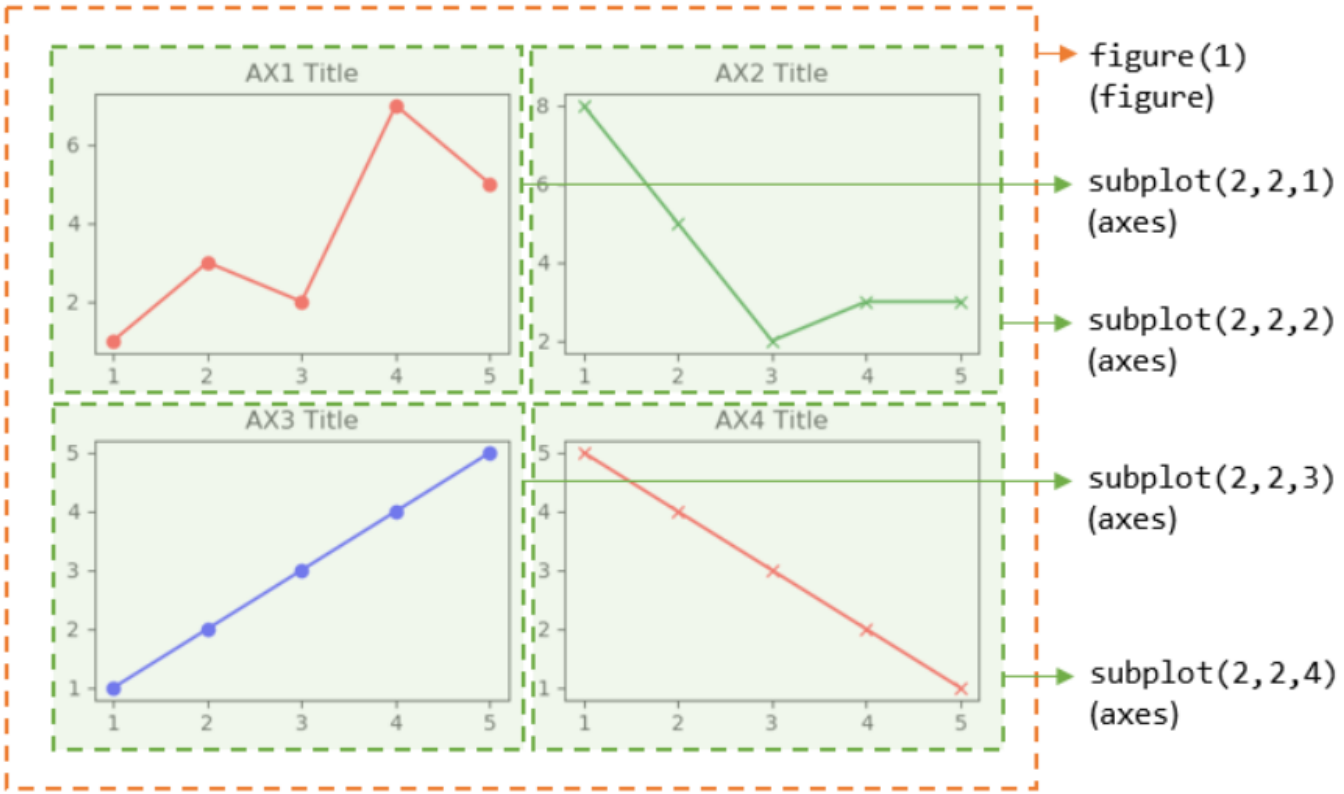
화면에 그리기

tutorial : https://matplotlib.org/stable/tutorials/introductory/pyplot.html

API : https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.html

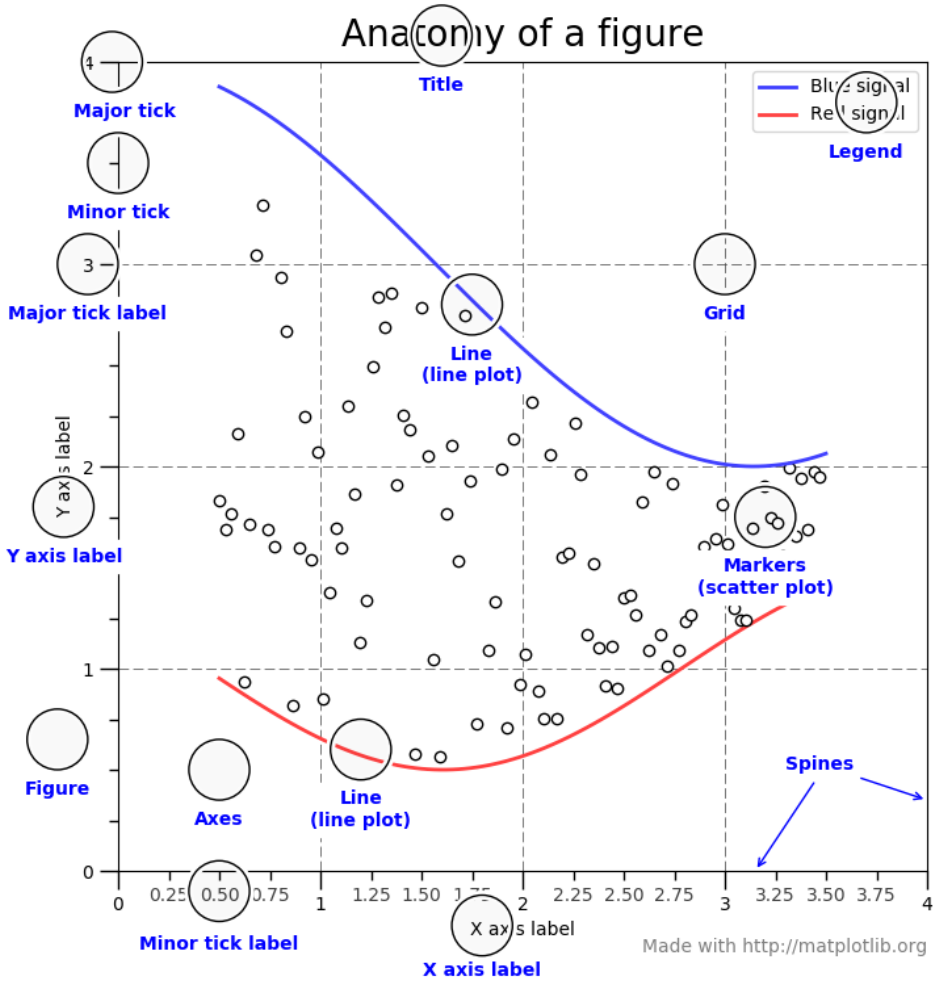
Plot 컴포넌트

여러 Subplot을 그리드로 배열하기




```
fig1, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
```

용어

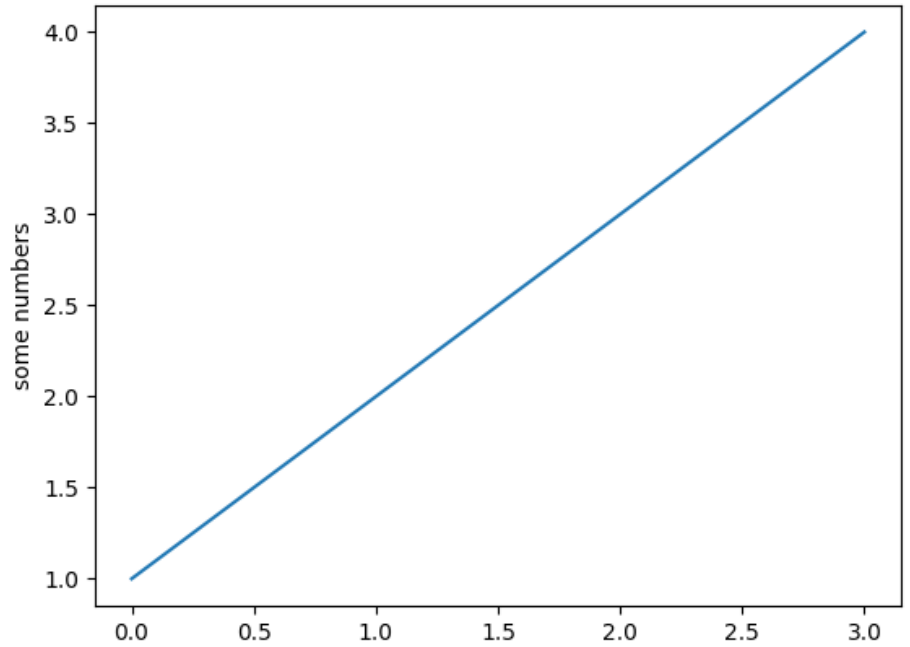


1. Figure : 그래프 전체 그림
2. Axes : 그래프가 그려지는 좌표 평면이자 subplot 단위
3. X axis : X 축
4. Y axis : Y 축
5. Tick : 눈금 (Major 눈금. Minor 눈금)
6. Spines : 그래프 테두리
7. Line : 선 그래프의 선
8. Markers : 그래프 상의 점의 형태
9. Grid : 그래프 격자
10. Title : 그래프 제목
11. Label : 각 축이나 눈금 등에 붙이는 문자
12. Legend : 범례 (그래프를 구분하는 이름 또는 설명 목록)

기본 그래프

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.show()
```

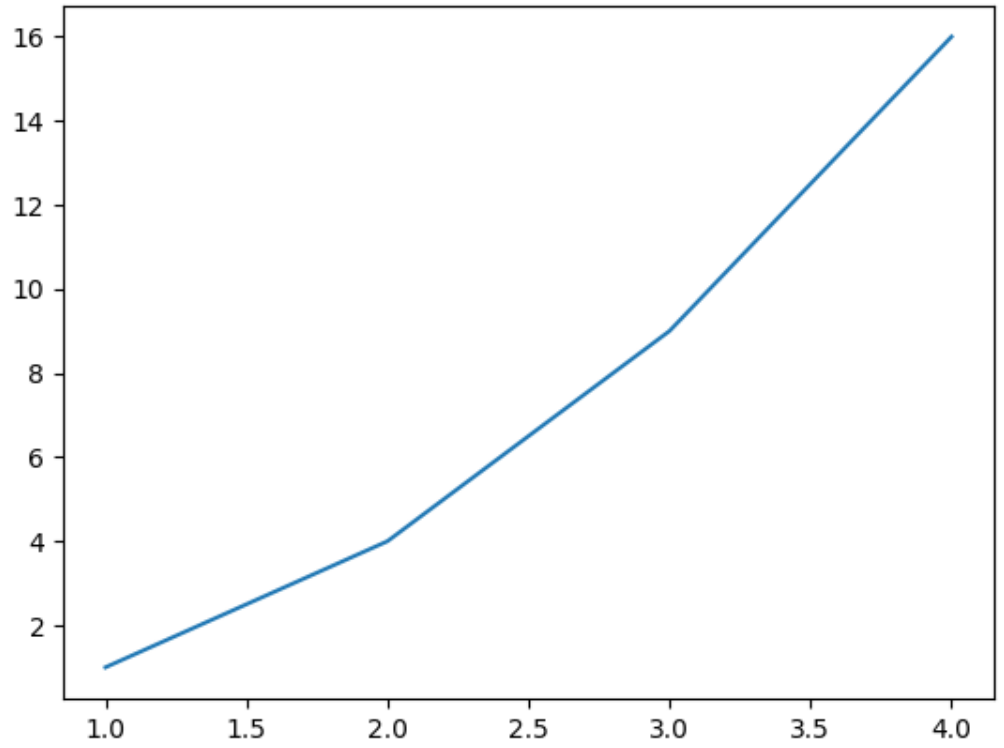
- Plot 함수의 리스트가 1개면 y 값으로 간주



기본 그래프

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
```

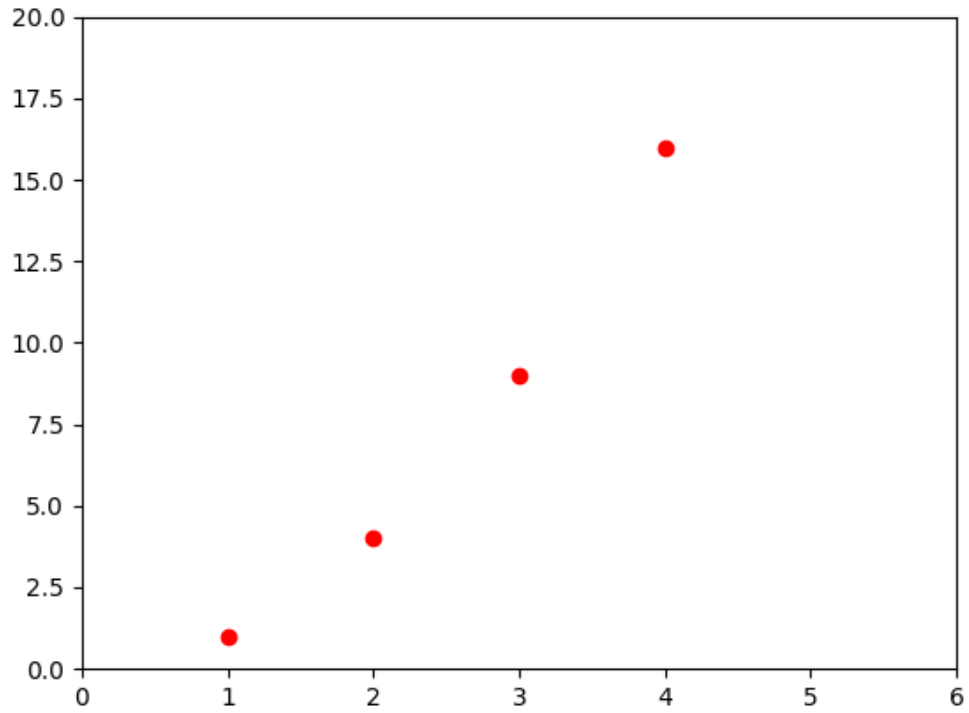
- Plot 함수의 리스트가 2개면 x값과 y값으로 간주



가로 세로 축 범위

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```

- `axis : [xmin, xmax, ymin, ymax]`



스타일 추가

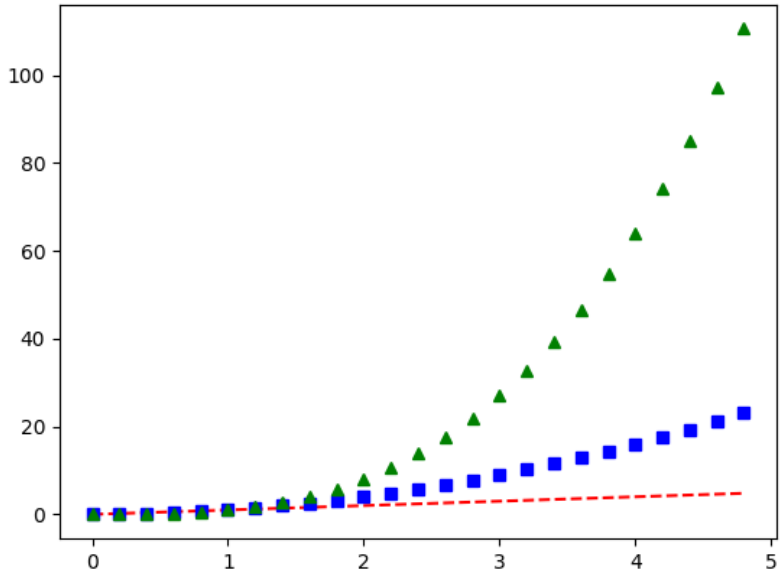
색깔 :
- 'r' (red), 'g' (green), 'b' (blue)
- 'c' (cyan), 'm' (magenta), 'y' (yellow)
- 'k' (black), 'w' (white)

마커:
- '.' (point marker), ',' (pixel marker), '*' (star marker), '+' (plus marker), 'x' (cross marker)
- 'o' (circle marker), 's' (square marker), 'h' (hexagon1 marker), 'H' (hexagon2 marker), 'd' (thin-diamond marker), 'D' (diamond marker)
- 'v' (triangle-down marker), '^' (triangle-up marker), '<' (triangle-left marker), '>' (triangle-right marker)
- '1' (triangle-down marker), '2' (triangle-up marker), '3' (triangle-left marker), '4' (triangle-right marker)

라인 스타일:
- '-' or 'solid'
- '--' or 'dashed'
- '|' (vline marker), '_' (hline marker)
- '-.' or 'dashdot'
- ':' or 'dotted'

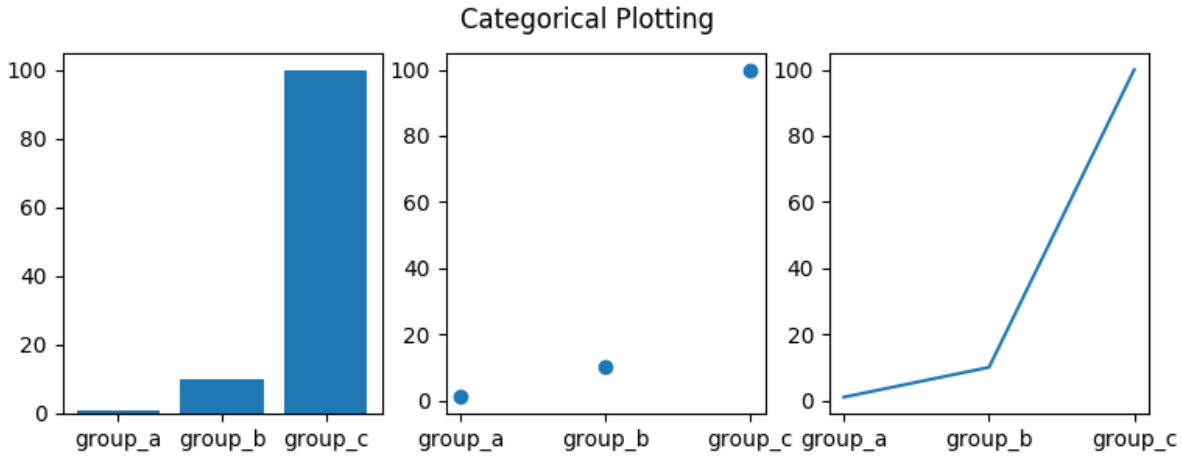
스타일 추가

```
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



범주형 데이터

```
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.bar(names, values)
plt.subplot(132)
plt.scatter(names, values)
plt.subplot(133)
plt.plot(names, values)
plt.suptitle('Categorical Plotting')
plt.show()
```

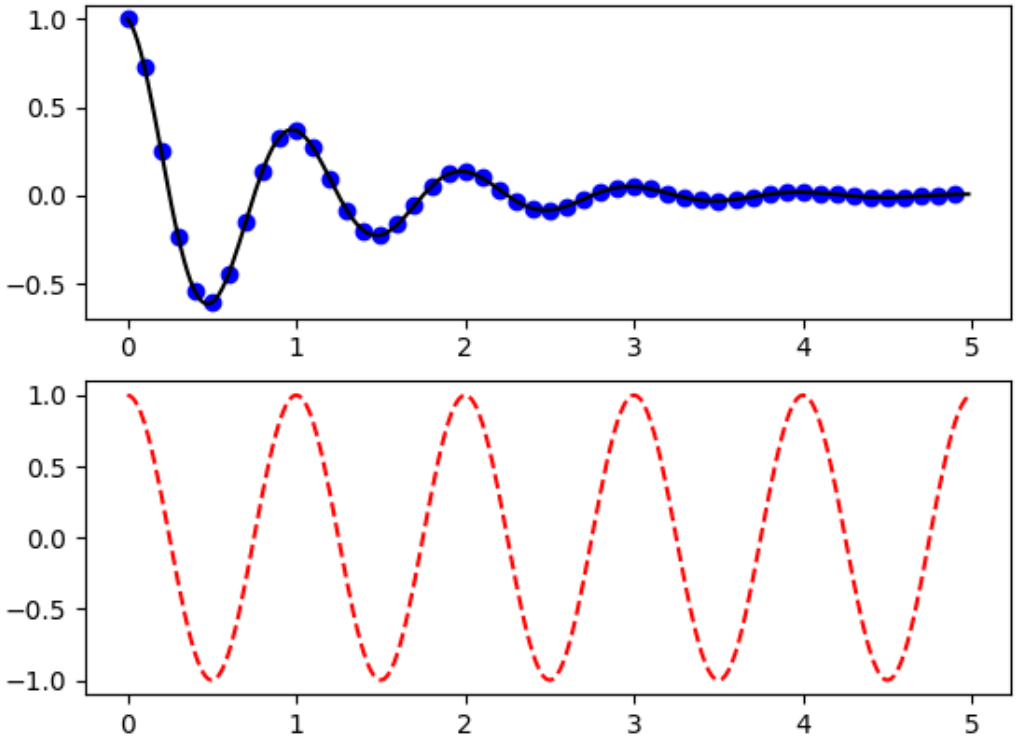


https://matplotlib.org/tutorials/introductory/pyplot.html

연속형 데이터

```
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure()
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```

$$f(t) = e^{-t}\cos(t * 2\pi)$$



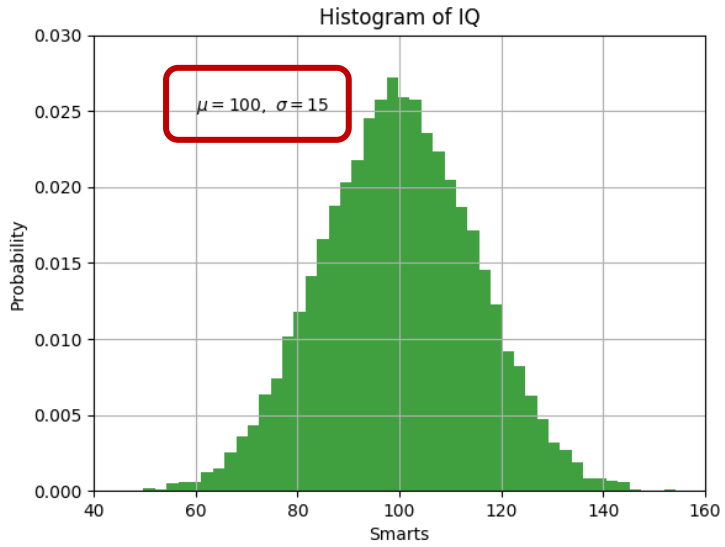
텍스트

```
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1, facecolor='g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

투명도

Bin의 개수

확률 분포로 normalize할 지 여부

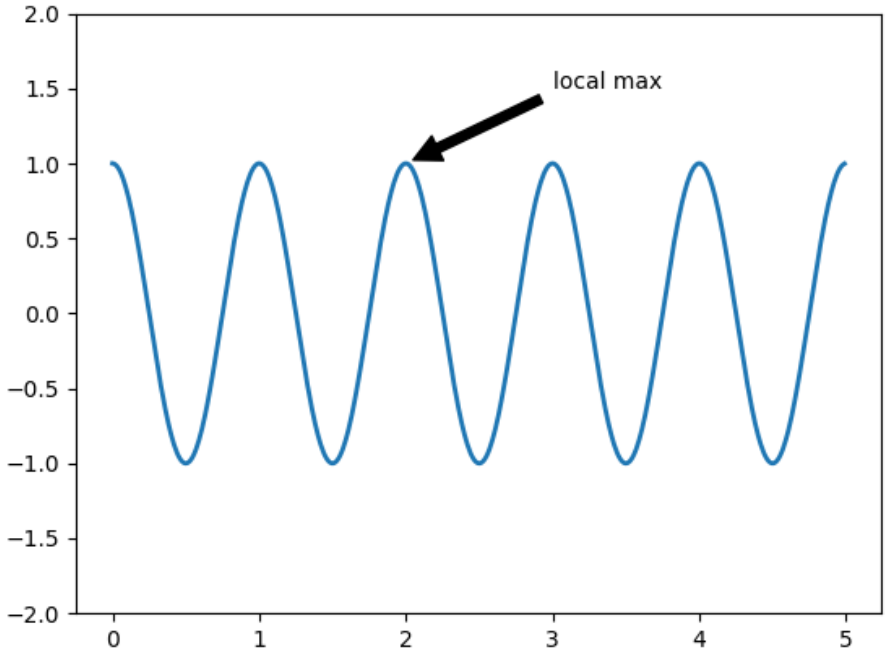


- 수식 표현에는 TeX equation expression을 사용

어노테이션

Annotating text

```
ax = plt.subplot(111)

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)
                       lw : linewidth

plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
                  arrowprops=(facecolor='black', shrink=0.05), )


plt.ylim(-2, 2)
plt.show()
```



annoate 함수

- xy : annotation 될 위치
- xytext : 텍스트 위치

이미지

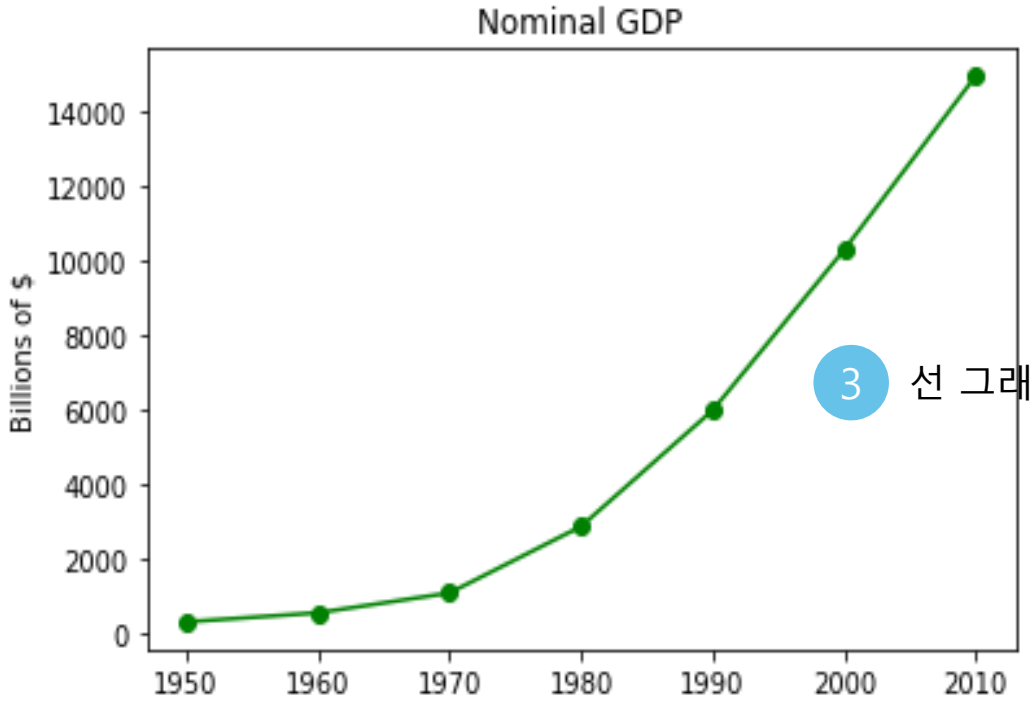
```
import matplotlib.image as mpimg

 # rescale to between 0 and 1
img = mpimg.imread("test_image.jpg") / 256
plt.imshow(img)
```

3. 예시

연도 별 GDP

연도별 GDP 데이터를 그래프로 어떻게 표현할 것인가?

```
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [300.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3]
```



2 세로 축은 GDP를 표현

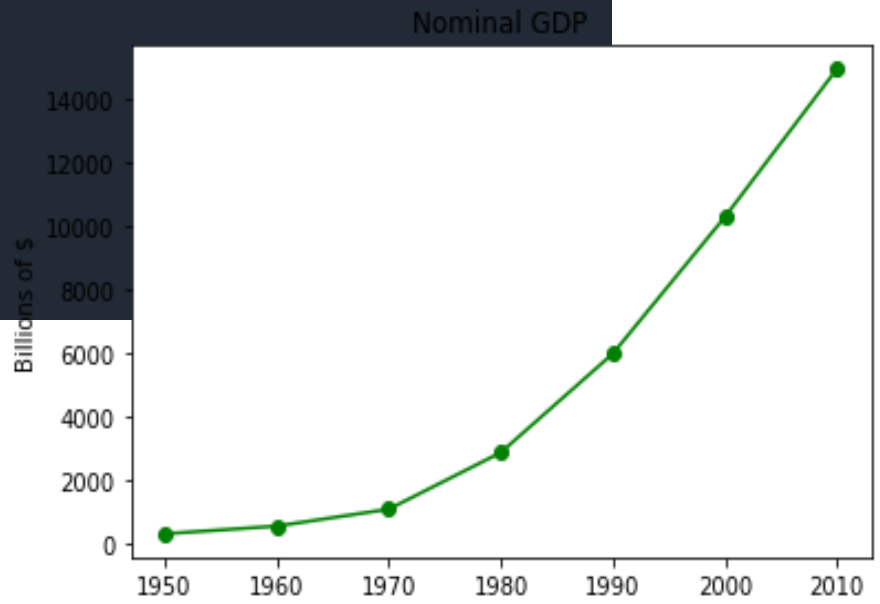
3 선 그래프로 표현

1 가로 축은 연도를 표현

선 그래프

연속된 숫자 데이터는 선 그래프로 표현한다.

```
from matplotlib import pyplot as plt

years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [300.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3]

# create a line chart, years on x-axis, gdp on y-axis
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')

# add a title
plt.title("Nominal GDP")

# add a label to the y-axis
plt.ylabel("Billions of $")
plt.show()
```



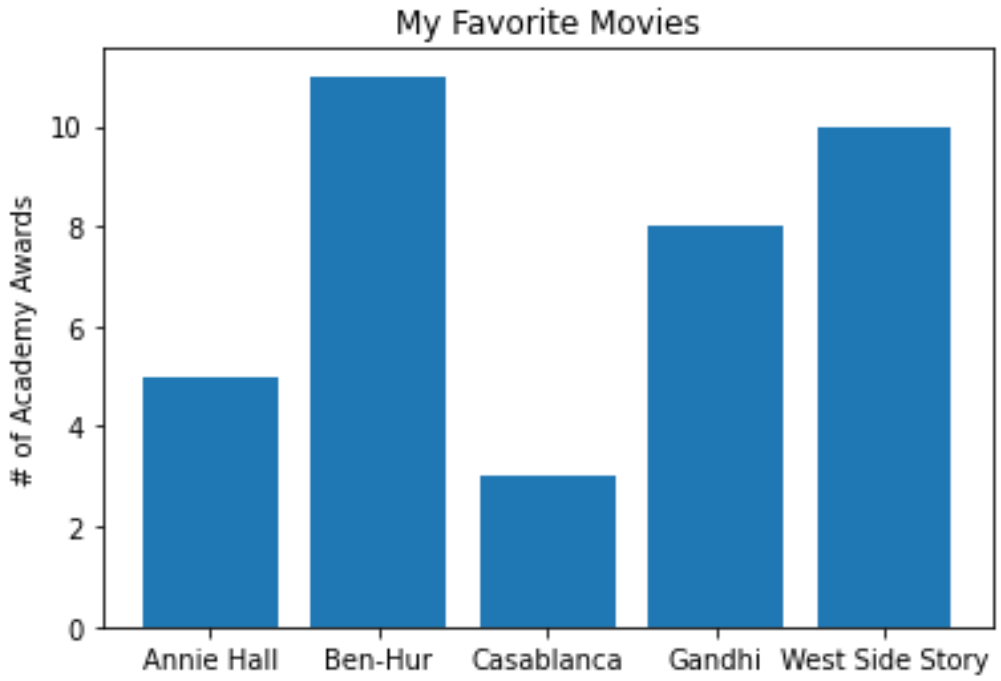
- title(label) : 그래프 제목
- ylabel(ylabel) : y축 레이블

오스카 상 수상 횟수

각 영화가 오스카 상을 수상한 회수를 표현하려면?

```
movies = ["Annie Hall", "Ben-Hur", "Casablanca", "Gandhi", "West Side Story"]
num_oscars = [5, 11, 3, 8, 10]
```

2 세로 축은 오스카 상 수상 횟수를 표현



3 막대 그래프로 표현

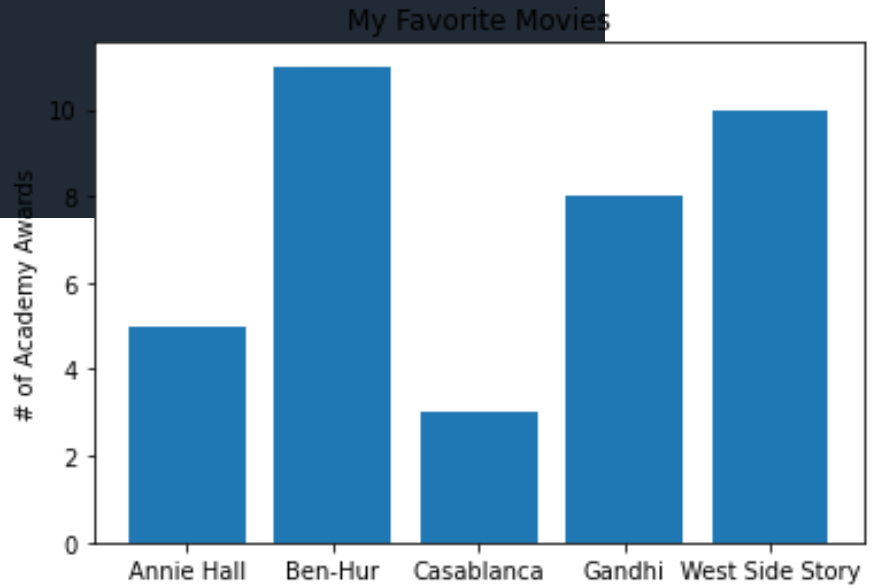
1 가로 축은 영화 제목을 표현

막대 그래프

범주형 데이터는 막대 그래프로 표현한다.

```
movies = ["Annie Hall", "Ben-Hur", "Casablanca", "Gandhi", "West Side Story"]
num_oscars = [5, 11, 3, 8, 10]

# plot bars with left x-coordinates [0, 1, 2, 3, 4], heights [num_oscars]
plt.bar(range(len(movies)), num_oscars)

plt.title("My Favorite Movies")     # add a title
plt.ylabel("# of Academy Awards")   # label the y-axis

# label x-axis with movie names at bar centers
plt.xticks(range(len(movies)), movies)

plt.show()
```

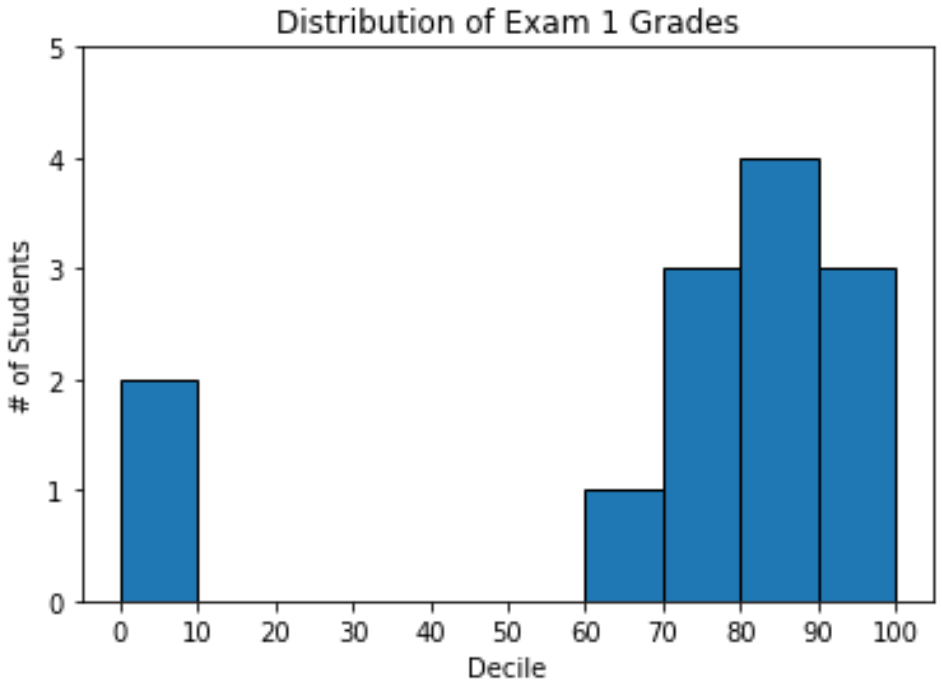
- bar(x, height) : bar 그래프
- xticks([ticks, labels]) : x축 눈금 (위치, 레이블)



성적

성적을 10점 단위로 묶어서 히스토그램으로 그려보자.

```
grades = [83, 95, 91, 87, 70, 0, 85, 82, 100, 67, 73, 77, 0]
```



2 세로 축은 성적 개수

3 막대 그래프로 표현

1 가로 축은 10점 단위로 성적을 나타냄

히스토그램

막대 그래프로 히스토그램을 표현할 수 있다.

히스토그램 데이터 만들기

```
from collections import Counter
grades = [83, 95, 91, 87, 70, 0, 85, 82, 100, 67, 73, 77, 0]

# Bucket grades by decile, but put 100 in with the 90s
histogram = Counter(min(grade // 10 * 10, 90) for grade in grades)
```

- 점수를 10점대로 나눠서 [0, 10, 20, 30, ..., 90] 구간으로 이산화한다.
- 단, 100점은 90점 대에 포함되도록 예외로 처리한다.

막대 그래프 그리기

```
plt.bar([x + 5 for x in histogram.keys()],  # Shift bars right by 5
        histogram.values(),                 # Give each bar its correct height
        10,                                 # Give each bar a width of 10
        edgecolor=(0, 0, 0))                # Black edges for each bar
```

- 막대 그래프 중심 위치가 5, 15, 25, ...가 되도록 5만큼 이동시킴
- 폭은 10으로, 테두리는 검정색으로

히스토그램

y축과 y축 범위 지정

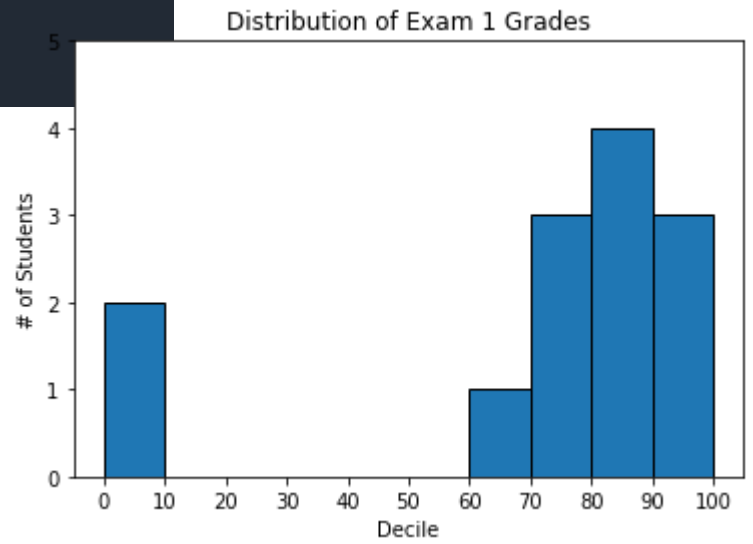
```
plt.axis([-5, 105, 0, 5])                    # x-axis from -5 to 105,
                                             # y-axis from 0 to 5
```

- x축은 -5부터 시작하도록 하며 105까지 표현하여 좌우 여백을 주도록 한다.
- Y축은 [0,5] 구간으로 설정한다.

눈금 및 레이블 지정

```
plt.xticks([10 * i for i in range(11)])    # x-axis labels at 0, 10, ..., 100
plt.xlabel("Decile")
plt.ylabel("# of Students")
plt.title("Distribution of Exam 1 Grades")
plt.show()
```

- x축 눈금은 10 단위로 표기 [0, 10, 20, …, 90, 100]

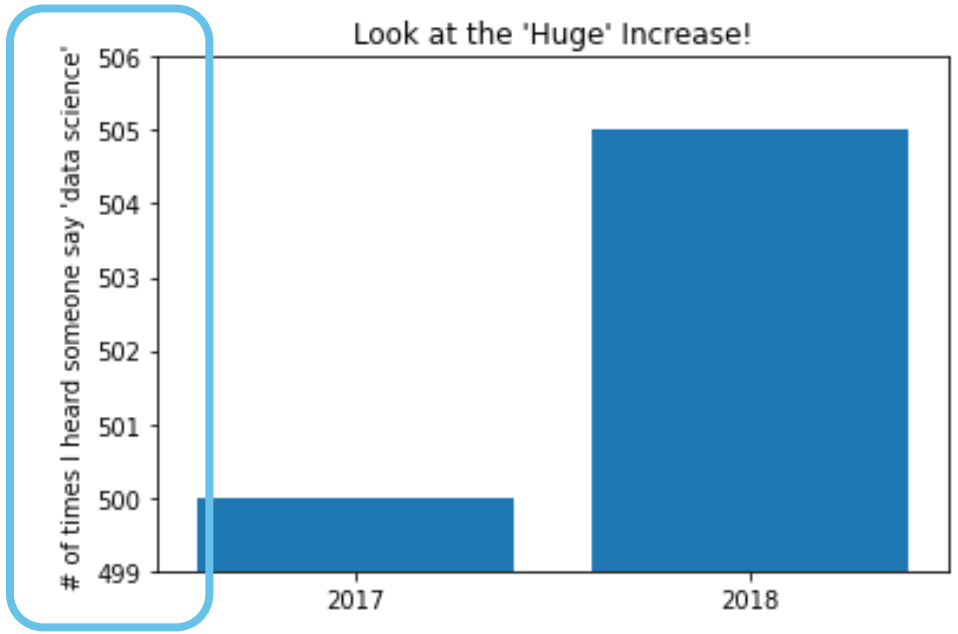


좌표 범위와 결과의 왜곡

좌표 구간을 유의미하게 지정하지 않으면 데이터가 과장되어 표현될 수 있다.

```
mentions = [500, 505]
years = [2017, 2018]
```

Matplotlib가 y축의 범위를 [499, 506]로 정해서 표현



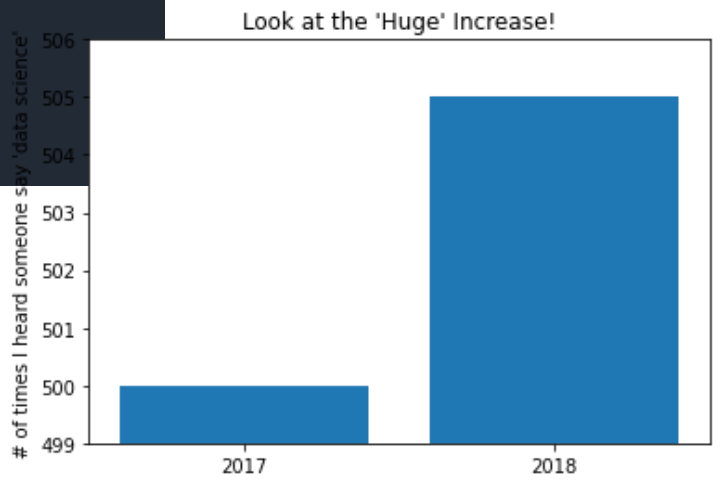
상대적으로 2018년도 자료가 아주 높게 보이게 됨

좌표 범위와 결과의 왜곡

Matplotlib가 자동으로 지정하는 범위를 확인하고 조정해보자!

```
mentions = [500, 505]
years = [2017, 2018]

plt.bar(years, mentions, 0.8)
plt.xticks(years)
plt.ylabel("# of times I heard someone say 'data science'")

# if you don't do this, matplotlib will label the x-axis 0, 1
# and then add a +2.013e3 off in the corner (bad matplotlib!)
plt.ticklabel_format(useOffset=False)

# misleading y-axis only shows the part above 500
plt.axis([2016.5, 2018.5, 499, 506])
plt.title("Look at the 'Huge' Increase!")
plt.show()
```

- ticklabel_format(₩*[, axis, style, ...]) : x, y축의 눈금 레이블 포맷지정
- Useroffset=Flase로 주면 오프셋 표기를 하지 않도록 설정하는 것



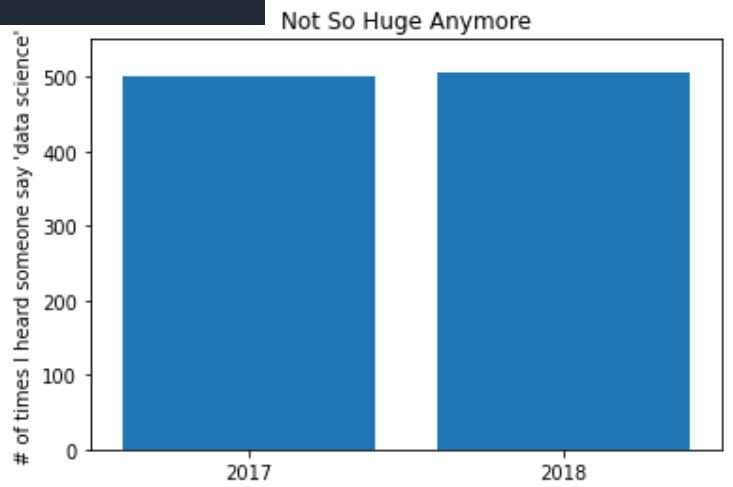
좌표 범위와 결과의 왜곡

범위를 재지정해서 결과가 큰 차이가 없음을 확인할 수 있도록 함

```
plt.axis([2016.5, 2018.5, 0, 550])
```

```
plt.bar(years, mentions, 0.8)
plt.xticks(years)
plt.ylabel("# of times I heard someone say 'data science'")
plt.ticklabel_format(useOffset=False)

plt.axis([2016.5, 2018.5, 0, 550])
plt.title("Not So Huge Anymore")
plt.show()
```

- axis(*args, **kwargs) : 좌표 축의 속성을 지정하거나 읽음

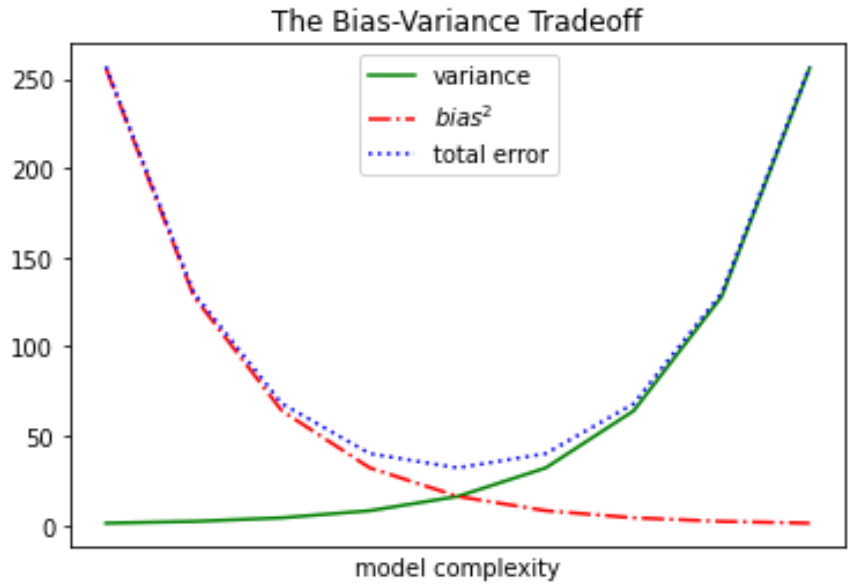


분산과 편향 그래프

분산과 편향, 전체 오차의 그래프를 합쳐서 표현해보자

```
variance     = [1, 2, 4, 8, 16, 32, 64, 128, 256]
bias_squared = [256, 128, 64, 32, 16, 8, 4, 2, 1]
total_error  = [x + y for x, y in zip(variance, bias_squared)]
```

2 세로 축은 GDP를 표현



3 선 그래프로 표현

1 가로 축은 연도를 표현

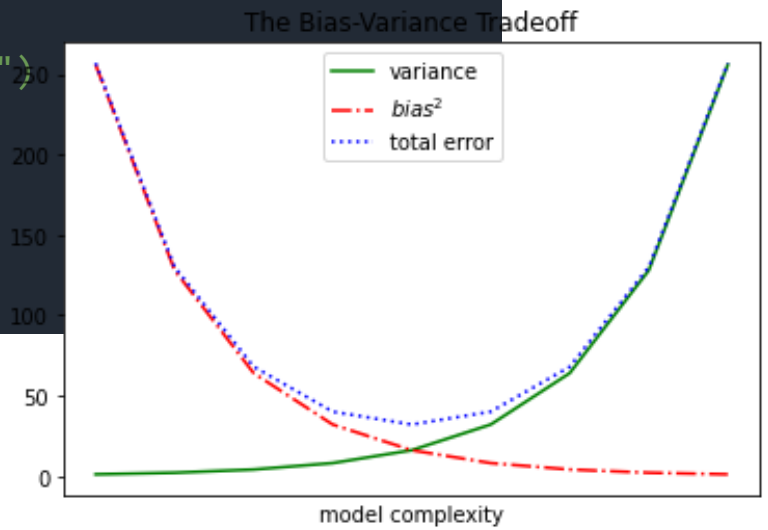
여러 그래프 합쳐서 그리기

plt.plot으로 그래프를 여러 번 그리면 하나로 합쳐서 그린다.

```
variance     = [1, 2, 4, 8, 16, 32, 64, 128, 256]
bias_squared = [256, 128, 64, 32, 16, 8, 4, 2, 1]
total_error  = [x + y for x, y in zip(variance, bias_squared)]
xs = [i for i, _ in enumerate(variance)]

# We can make multiple calls to plt.plot
# to show multiple series on the same chart
plt.plot(xs, variance,     'g-',  label='variance')    # green solid line
plt.plot(xs, bias_squared, 'r-.', label='$bias^2$')      # red dot-dashed line
plt.plot(xs, total_error,  'b:',  label='total error') # blue dotted line

# Because we've assigned labels to each series,
# we can get a legend for free (loc=9 means "top center")
plt.legend(loc=9)
plt.xlabel("model complexity")
plt.xticks([])
plt.title("The Bias-Variance Tradeoff")
plt.show()
```

Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

- legend(*args, **kwargs) : 범례 그리기

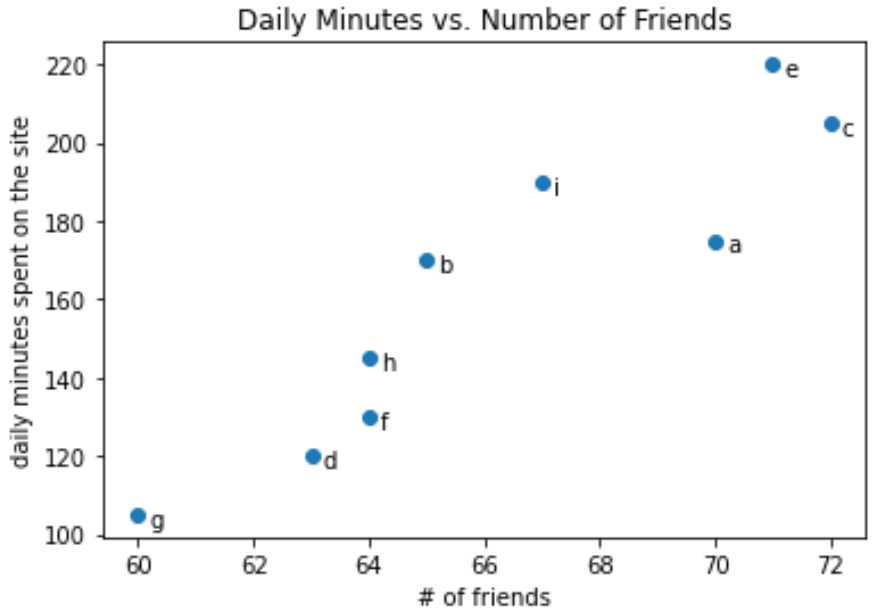


친구 수와 사이트 체류 시간

친구가 많을수록 사이트에 오래 머물러 있을까?

```
friends = [ 70,  65,  72,  63,  71,  64,  60,  64,  67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
labels =  ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
```

2 사이트 체류 시간



3 산점도로 표현하고
각 점마다 레이블을 표현

1 친구 수를 표현

산점도

두 변수의 연관 관계를 표현하려면 산점도로 표현하라.

산점도 그리기

```
friends = [ 70,  65,  72,  63,  71,  64,  60,  64,  67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
labels =  ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

plt.scatter(friends, minutes)
```

- scatter(x, y[, s, c, marker, cmap, norm, ...]) : x좌표, y좌표

각 점마다 레이블 붙이기

```
# label each point
for label, friend_count, minute_count in zip(labels, friends, minutes):
    plt.annotate(label,
        xy=(friend_count, minute_count), # Put the label with its point
        xytext=(5, -5),                  # but slightly offset
        textcoords='offset points')
```

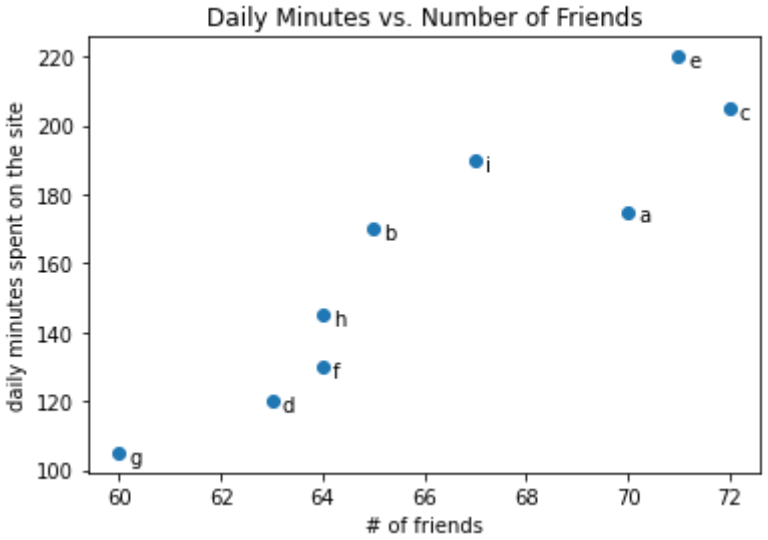
'offset points' Offset (in points) from the *xy* value

'offset pixels' Offset (in pixels) from the *xy* value

산점도

산점도 그리기

```
plt.title("Daily Minutes vs. Number of Friends")
plt.xlabel("# of friends")
plt.ylabel("daily minutes spent on the site")
plt.show()
```



좌표 범위와 결과의 왜곡

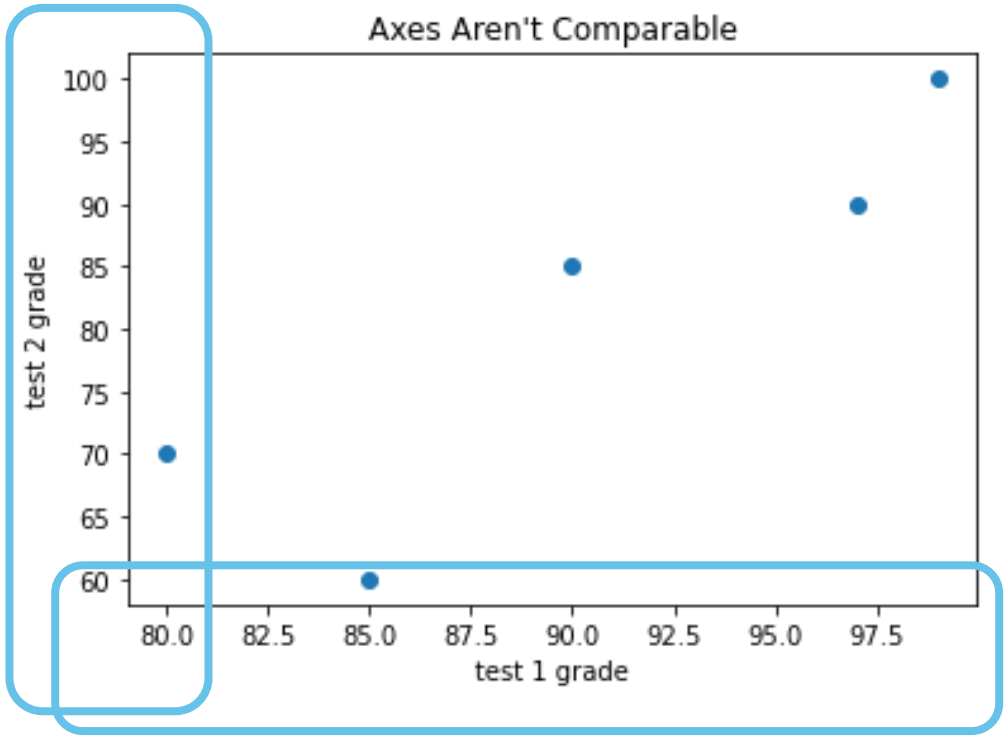
두 시험 성적 간의 관계를 분석한다면?

```
test_1_grades = [ 99, 90, 85, 97, 80]
test_2_grades = [100, 85, 60, 90, 70]
```

```
test_1_grades = [ 99, 90, 85, 97, 80]
test_2_grades = [100, 85, 60, 90, 70]

plt.scatter(test_1_grades, test_2_grades)
plt.title("Axes Aren't Comparable")
plt.xlabel("test 1 grade")
plt.ylabel("test 2 grade")
plt.show()
```

test_1의 편차가 test_2만큼 큰 것으로 표현이 됨

y축의 범위 : **[55, 105]**



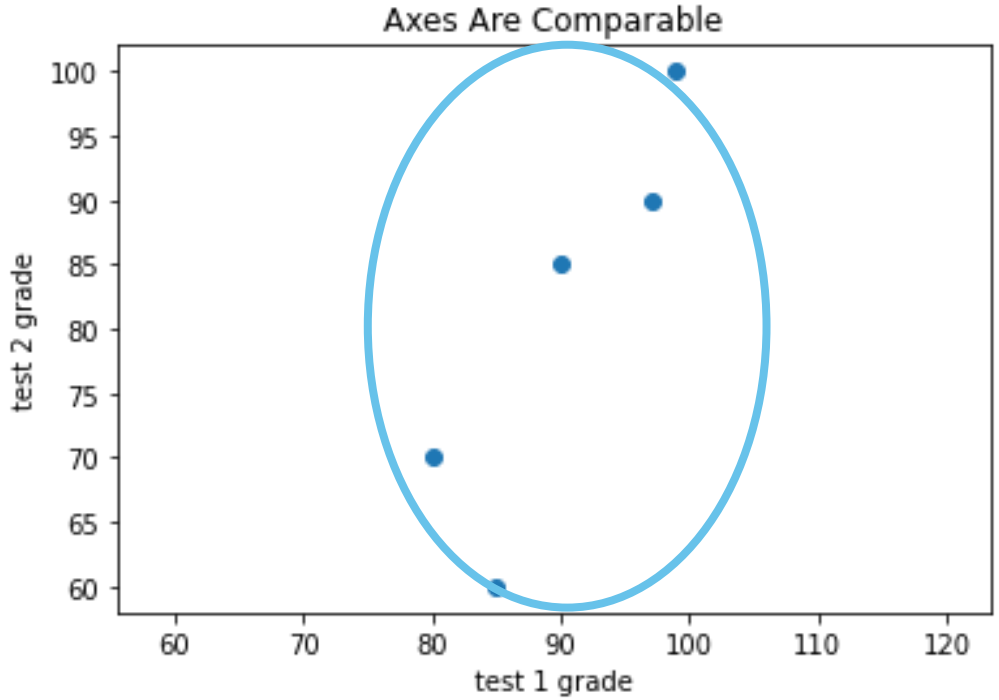
x축의 범위 : **[75, 100]**

좌표 범위와 결과의 왜곡

두 축의 범위를 같게 만들어서 편차의 크기를 공정하게 비교하도록 함

```
plt.axis("equal")
```

```
test_1_grades = [ 99, 90, 85, 97, 80]
test_2_grades = [100, 85, 60, 90, 70]
plt.scatter(test_1_grades, test_2_grades)
plt.title("Axes Are Comparable")
plt.axis("equal")
plt.xlabel("test 1 grade")
plt.ylabel("test 2 grade")
plt.show()
```



Thank you!